Serial Server Page

Local Station application

Oct 17, 1989

Overview

The Serial Server facility for the Fermilab D0 VME systems provides for a serial RS–232 connection to the Token Ring network of VME stations. A computer plugged into the serial port of any VME station can request data from any devices in the entire network of stations. In addition, settings can be made to any device in the network. These two commands are supported with the full generality inherent in the VME station software, the only real limitation being that of the serial baud rate. One can program any application in a familiar environment and yet have access to all the data in the system.

The Serial Server is implemented as an application program running on a VME station. The station used does not have to be the same station providing the serial port, although that might be a likely. Since it is an application, however, the server functions will stop if one leaves the application page. Currently there is a limitation of six simultaneously active requests supported by the server. A request for data can be a one-shot request, or it can be repetitive. The maximum repetition rate is 15 Hz, although one may reach the baud rate limit soon at that rate if too much data is being requested. The serial data is returned via a spooling mechanism using the available dynamic memory, so that the full baud rate can be delivered.

Message encoding

The command messages used to request data and make settings are of the same format as is used by the Token Ring network, but it is expressed in an ASCII encoding method based upon the Motorola `S`-record object format. That format provides for an `S0` header record, followed by any number of `S1` data records comprising the content of the block, followed by an `S9` record which terminates the block. Each record includes a length byte, a two-byte "Load Address" (LA), any content bytes, and a checksum byte, all expressed in hexadecimal ASCII at two characters per binary byte of data. (Note that hex digits `A–F` are expressed in upper case.)

The `S0` header record LA is used in a *response* to indicate an error when nonzero. The header content data bytes allow the user system to tag the request with a request-id, which may be used to match the returned data block with the request. The `S9` terminating record uses the load address field to specify the number of `S1` data records included in the block. The receiver can test this count to insure that no data records were lost in the block transmission. In order to use a format which is *incompatible* with the Motorola format, the non-hex character `S` is expressed as a lower case `s`. This is to insure that such serial records do not mistakenly get used as input to the Download application program.

Examples

An example of a request for a single reading of channel `04:000` (channel 0 in system 4) to be returned at a 1 Hz rate is as follows: (Note that the spacing indicated here is to aid the eye for illustration only.)

```
s0 05 0000 0001 xx<CR>
s1 0D 0000 2001 0F 01 2001 0002 0400 xx<CR>
s9 03 0001 FB<CR>
```

The first record specifies a request-id of `0001`—it may be any value up to 10 bytes long. The second record comprises the entire request command. The `2001` identifies the command as a request-for-data using list 1, the `0F` is the count of 15 Hz cycles representing 1 Hz repetition rate, the `01` is the number of listypes (which specify the type of data requested), the `2001` specifies 2 bytes per ident and only one ident in the request, the `00` is listype 0 (A/D readings), the `02` means that two bytes are requested, and the `0400` is the (short) ident for channel 0 of system 4. The third record includes the `0001` to indicate the count of `s1` records in the block.

The response to be expected from this command might be as follows:

```
s0 05 0000 0001 xx<CR>
s1 09 0000 0001 0000 4000 xx<CR>
s9 03 0001 FB<CR>
```

The header record specifies `0000`, signifying no errors were detected in the previous request command. (If the request command was received and recognized, but something was wrong with it, the reply would have consisted only of an `s0` record with a nonzero LA field.) The `0001` echoes the request-id sent in the request command. The `s1` record has a `0001` to indicate that it is an answer message (the hi nibble=0), and that the list number =1. The next `0000` word means no errors from the data collection system. The word of `4000` is a hypothetical reading value, which would represent here one-half of full-scale, or 5 volts at the A/D input. The `s9` record again provides the `s1` record count.

In almost all of the above records, an `xx` is shown to signify the checksum byte. It is computed by summing the binary bytes (not the ASCII character codes) beginning with the record length byte and ending with the last data byte, and one's complementing the result to form the byte which is then ASCII-encoded. (As an example, the checksum for the `s9` record above is `FB`.) Notice that the receiver when checking for checksum errors merely has to sum all the bytes mentioned above plus the received checksum byte. If the result is `FF`, the checksum is correct.

An example of a setting command might be to set channel 7 in system 4 to one volt. The setting command would look like this:

```
s0 05 0000 0056 xx<CR>
s1 0B 0000 3001 01 02 0407 0CCC xx<CR>
s9 03 0001 xx<CR>
```

The `0056` value in the `s0` record is the arbitrary request-id. The `s1` record shows a `3001`, which signifies a setting command, with an ident length of one word. The `01` byte is the setting listype #, and the `02` means we are sending two bytes of data. The `0407`, of course, represents the channel ident (short form) specifying channel 7 in system 4. The data value is `0CCC`, which is one-tenth of full scale, or 1 volt. The `s9` record is as before.

Serial Server Application

Here is the application page format of the Serial Server as displayed on the small consoles of the VME systems:

```
S SERIAL SERVER   01/31/87 1332
INPUT SYS:2     #RECORDS=    21
*ACTIVE            #ERRS=     0
 L  REQS    NB ANSWERS    NB  E
*1    12    36    1230    16
 2     1   144       1   308
 3
 4
 5
 6

S00500000003xx                        □
S10D000020020001200100020700xx        □□ data request
S9030001FB                            □
S1090000000200005678xx                □ response
```

(This sample is purely illustrative and does not represent completely consistent example data values.)

Upon entry to the page, the second line indicates `*ACTIVE`. To change the node whose serial port is used for the serial server, enter the system number of the station which is attached to the serial port (if changed), and with the cursor on the 2nd or 3rd line, press the interrupt button. The second line will indicate `*ACTIVE` as above and will remain so until one leaves the page, which will terminate operation of the server. The rest of the page merely shows diagnostics of the server operation.

The number of records received from the user system and the number of error records detected is shown. The table shows diagnostic values for each of the six possible active requests which can be handled by the server. The * before the request number indicates that the request is currently active and collecting and delivering answers. The second field shows the number of separate request commands received using the given list number. The next field is the number of bytes (not ASCII characters) comprising the request itself. The next field counts the answer blocks returned, and the next one is the number of bytes in the returned answers. The last field shows any error status of the data collection system.

The last lines display a snapshot of the latest request that was received. The first 3 show the header record, the first data record, and the terminating record of the request. The last record shows the first data record of the most recent answer response, which will continue to update until a new request is received, when these 4 lines will change to give diagnostics for that request.

Baud Rate Timing

To get an idea of what the baud rate timing limitation is, assume that 9600 baud is used, and we request the single word of data as described in the first example. The number of ASCII characters is the answer response is 52, assuming that `<CR>` represents both a carriage return and a linefeed character. At about 1 ms per byte, this amounts to 52 ms of serial time—quite busy at a 15 Hz rate but rather comfortable at 1 Hz. Further, if we requested a set of 20 readings—again originating from any stations in the network—the time would be about 140 ms, assuming that the response block contained two `s1` records. And if we requested 100 words, it would require about 500 ms. Of course, operation at 19.2K baud would halve these figures. Therefore, we have that the serial time for n data words=

(48+4.6n) ms @ 9600 baud

(24+2.3n) ms @ 19.2K baud